8.1 Document essentials

Anyt ing t at is ritten a out o a system is designed or unction is documentation

Characteristics of A Good Documentation: DO A

Documentation is considered to be good if it has the following qualities:

a. Availability: It should be accessible to those for whom it is intended.

b. Objectivity: It must be clearly defined in a language that is easily understood.

c. Cross-referable: It should be possible to refer to other documents.

d. Easy to maintain: When the system gets modified, it should be easy to update the documentation.

e. Completeness: It should contain everything needed, so that those who have gone through it carefully can understand die system.

Need for documentation:

Preparation of documentation is quite important as it depicts what the system is supposed to be and how it should perform its functions. It illustrates both technically and economically how a system would better serve the objectives and goals of the company. Documentation improves overall operation in addition to management and audit control. The purposes for documentation are :

1. Without proper documentation, effective communication of who, what, where, when and how of the system is difficult.
2. Audit ability and Control are difficult to achieve without documentation.
3. If changes are made then it will not be clear to those who try to comprehend the system later on.
4. While converting from one system to another system if there is no proper documentation, there will be chaos.

It also serves the following purposes:

i. Reviews the progress or development of application software.

ii. Communicates facts about system to users.

iii. Communicates between personnel working on a development project.

iv. Provides necessary guidelines to allow correction or revision of a system or its computer programs.

v. Provides operating instruction to users and operating staff.

vi. Assists in the reconstruction of the system in case it is destroyed.

vii. It helps the management to determine if the new design achieves the objectives of the company within the established constraints and if it is justifiable from a cost standpoint.

viii. Documentation serves as a focal point from which the analysts' design can be assessed and as a standard to be utilized as a reference once the system is implemented.

8.2 Documentation Methods

Each phase in the systems development cycle is accompanied by appropriate documentation. The systems request, even if it is initially mark verbally, eventually must be written. It is desirable for the client and a systems analyst to work jointly in writing the request since each can contribute knowledge the other does not have. The written system's request is merely a statement of the user's problem.

Documentation also includes plans to test the system and convert from the old to the new one. The systems analyst must also provide a plan to train the personnel affected by the changes. During the life cycle of the completed system, the system itself must provide documentation of how well it is operating and consequently should be designed to yield data about itself as a normal by-product.

Anything that is written about how a system is designed or functions is documentation. Various documentation methods are: **1)System Documentation, 2)Programming Documentation, 3)Operations Documentations, 4)User Documentation(Developers & User Manuals), 5)Management Documentation, 6)Training Documentation, 7)Implementation Plan, 8)Appendix**

1.System documentation:

It describe the overall system design and include system flowchart, input/output formals, file descriptions, control requirements, report specifications etc. For example it may contain,

I.

a) Name of the system.
b) Why the system was developed.
c) Purpose and objectives.
d) Who are the users?
e) Where the system fits in the Company.
f) Description of design methodology (Training or Structured).

II

A) Narrative description.
B) Structured documentation
  1. Context DFD.
  2. Logical DFD.
  3. Level 0 of Physical DFD.
  4. Lower level DFDS.
  5. Data structure and data access diagrams.
  6. Data dictionary.
  7. Mini specifications for processes including logic
    i) Decision Tables.
    ii) Decision Trees.
  8. System Structured Charts.
C) Traditional documentation
  1. System Flowchart.
  2. Documented Flowchart and Documentation section.
  3. System requirements specification.
  4. Input/Output sheets.
  5. Equipment sheets.
  6. Personnel sheets.
  7. File sheets.
  8. Lay out Charts and Cost sheets.
D) Control's Matrix and Auditor's Report.

2. Programming documentation:

Before a program is developed, the systems analyst should provide the programmer with the required documentation. The logic in some programs is best described by a flowchart. Sometimes, decision tables are most appropriate for explaining the logic of a program. Programmers should insist on proper documentation before starting a job.

This includes programming specifications, descriptions of program logic including graphics aids such as program flowcharts. Design and program documentation are for the use by technical personnel. A more detailed list is give here:

1. Program Listing.
2. Comments and Notes Statements.
3. Structured Charts.
4. One Program Flowchart for each program and an overall flowchart showing how the various programs interact. (System Structure Charts).
5. Detailed narrative description and decision trees or decision tables for complicated logic or calculations within each program (mini specifications).
6. Pictorial Layouts of files, the outputs and inputs (Data Structure Diagram, Data Access Diagrams).
7. Program Test Plans and Results.
8. Copying in final form of all input/output documents affecting the program.
9. Statements of standards for coding structures and input/output layouts.
10. Clarification of the program's interface with other related programs.

The programmer's responsibility in documentation is to provide information to enable future programmers to make necessary changes. Personnel turnover is normal feature in any business, and turnover is particularly high among programmers. A company can never think that a programmer assigned to a specific program will be available in two years, when some modifications to that program are required. For continuity of information a company must insist on complete and meaningful documentation. Typically a documentation folder is provided for each program which contains all the input/output forms associated with the program, a detailed flowchart or decision table for the program use a set of operator and user instructions.

Maintaining this type of documentation is costly and time consuming, for, programmers do not take interest in spending time, for this type of work. Routine changes occur frequently in a program and all changes must be covered in the documentation folder. But the very changes which require the updating of existing documentation are the reasons for maintaining accurate documentation.

3. Operations documentation:
A well designed system may run for a long time with little or no assistance from the systems department. This can happen only when the system has been documented in a proper way. Operations documentation is for those who will keep the system running from day to day. It contains :

A) Operating instructions for normal operations.
B) Directions for handling problems and breakdowns.

For smooth running of the system, the console operator must have complete knowledge about the job. Providing the computer centre with a set of operating instructions will not serve the purpose. The instructions must be in a form readily accessible to the console operator and written in simple and understandable style. A systems analyst must thoroughly discuss all the requirements of new jobs with the operations staff before the job can be properly transferred.

The run book is traditional in computer centers. It is a collection of operator instructions for each program at an installation and typically contains:

i. Narrative, describing the run
ii. Listing of the programmed error conditions
iii. Detailed information for running the job, including:
   - input/output forms to be used
   - anticipated problem areas and how to handle them

- detailed description of file assignment of each input/output device
- disposition of data files after completing the job
- general block diagram of the programming logic
- restart procedures

The run book generally takes the form of a loose leaf notebook because of the case of substituting sheets as program change. It should be kept in mind that an operator in a multiprogramming environment must monitor many programs simultaneously. Instructions must be simple and complete enough for executing the job correctly.

4. User Documentation:
Systems users require proper documentation to prepare a developing system and to smoothly carry out existing ones. To meet this requirement, each system should have a manual that spells everything the users must know to do their job correctly. Users require two general types of information; complete details to handle each case the system processes, and overall picture of the system so that they can see their role in the total operation of the company. The manual should supply the following information.

- General flowchart of the system
- Assignment of responsibility for specific tasks
- Standards for work flow, including target dates and deadlines for specific tasks
- Simple input and output documents
- Detailed procedures
- Anticipated exceptions and instructions on how to handle them
- Accuracy standards for data in the system

The systems department must write a thoroughly detailed narrative of each system, including the proper handling of routine cases, as well as exception handling. A staff member in the user department must have an authority to consult when faced with a case not handled before. Properly prepared manual which is always available can provide the information needed by the user. Supervising staff in user areas must understand the overall picture in each system just as staff members must understand the details of their function. This requires documentation, in the form of charts, graphs and illustrations, so that the supervising staff has a clear grasp of their department's role in the total system.

5. Management Documentation:
The documentation required by corporate management differs quite a lot from that required by users. The systems designer must know the requirements of the management and provide documentation to enable management to perform three functions:

1. Evaluate progress on systems development
2. Monitor existing systems
3. Understand the objectives and methods of new and existing systems

Management is primarily interested to know in general the system's overall objectives and basic operations. A brief manual highlighting the key steps in each system may be prepared for management. Good managers have an exceptional ability to get to the root of a system and, their experience should enable them to retrieve information from a systems summary or chart which may not be apparent to the systems analyst.

6. Training Documentation:
includes the user training manuals and materials to be used in the conversion and installation of new system. User must know to fill out forms, how to correct errors and how to interpret output.

7. Implementation Plans:

Implementation plans and the results of implementation must be documented.

8. Appendix:

This contains all other documentation. For example,

A) Feasibility study report.

B) Problem definition report.

C) Study plan.

D) Normalization documents.

E) Summary of General information understanding existing system, new systems requirements , economic cost comparisons.

F) Final written report.

G) List of controls.

Thus it becomes evident that documentation

1. should be adequate
2. should be clear and
3. should be updated regularly.

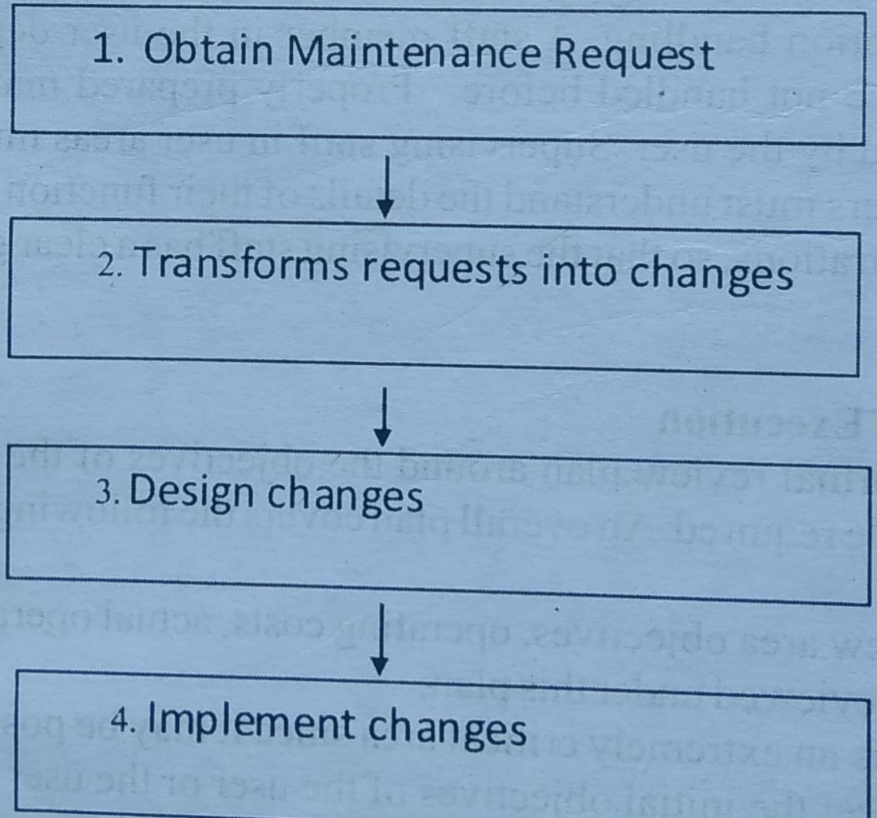
8.5 Application Change Management

Change management is the process of planning and coordinating the implementation of all changes – any planned alteration or addition to the operating environment including application systems – into the production environment in a logical and orderly fashion. It does not develop changes, nor does it implement them.

Importance of Maintenance:

When do we say that the system development is complete? When the system is operational we can say that the system development is almost complete. However, when the system is in operation, the most needed thing is its maintenance.

The maintenance may include hardware as well as software maintenance. In general the maintenance process can be stated in the form of a figure as given below:


1. Obtain Maintenance Request
2. Transforms requests into changes
3. Design changes
4. Implement changes

In short, System Maintenance involves maintenance that closely resembles mini systems development steps. The reason why maintenance is required is difficult to understand.

First, maintenance is required to keep software current with changing processing needs and requirements.

Secondly, maintenance is required to fix errors or bugs in the software. Even experience programmers realize that no software can be guaranteed to be free of bugs. A program may run successfully 99 times out of 100, only to fail on the 100th time.

Thirdly, the problem may arise in attempting to extent a system beyond its design limits. For example, an inventory system designed to handle 1000 items may be required to handle 1400 items.

Fourthly, computer technology goes on changing constantly. Maintenance is needed to keep pace with such changes.

Fifthly, maintenance is required to perfect or line tune coded procedures.

Types of Systems Maintenance:

There are four main types of systems maintenance. They are as follows:

1) Adaptive Maintenance
2) Corrective Maintenance
3) Perfective Maintenance
4) Preventive Maintenance

1) Adaptive Maintenance:-

This is performed with a view to keeping the software current. As internal and external business conditions change, the data-processing requirement also changes. Naturally, an organization must modify its operational software accordingly.

The adaptive maintenance begins with the request for service from the user. This may be assessed by a committee. Once approved, it goes to the maintenance programmers who treat this as a mini systems development assignment. The maintenance programmer must define system modification requirements, after the record modifications to terms used in constructing system diagrams and input/output formats.

In addition, adaptive maintenance also includes other housekeeping tasks. This is done with the help of housekeeping utility programs which include file copying and back-up, purging in active customers, file reorganization, file conversion, table update to include employees pay rates, revised tax rates, depreciation rates etc.

2) Corrective Maintenance:

This is performed in response to software failure. The errors that occur can be classified as:

1. Program logic errors such as inadequate data validation procedures, data types error, inconsistent use of variable names, incomplete logical paths etc.
2. System errors may consist of hardware problem or with software design.
3. Operations error can be traced to computer operations, computer schedules and members of the operating staff. The errors may be loading incorrect tap file, improper back up of computer files, failure to change printer ribbons etc.
4. User errors can be directly traced to user groups. These errors suggest deficiencies in training demonstration and user guides.

A solution to this is in form of module repairing which is the redesign of error prone modules in an implemented software system.

3) Perfective Maintenance:

This is performed to improve or maintain program efficiency. Modifying program data structures is one way to improve overall program efficiency. Backing records, storing precomputed results, and eliminating temporary work files are examples of ways to shorten processing run-times. Another way to improve efficiency is to modify expensive parts of a system, improving loops, test comparisons; calls to external procedures and evaluation of algebraic expressions illustrate ways of improving overall of processing.

By working more on adaptive and perfective maintenance tasks and less on corrective maintenance programmer strikes toward corrective the ideal.

4) Preventive Maintenance:

Similar to perfective maintenance, preventive maintenance involves changing some aspect of the system to prevent failures. It may include the addition of type checking, the enhancement of fault handling, or the addition of a "catch-all" statement to case statement, to make sure the system can handle all possibilities. Preventive maintenance usually results when a programmer or code analyzer finds an actual or potential fault that has not yet become a failure and takes action to correct the fault before damage is one.